Homework 1

Zi Han Zhao

1001103708

1 Longest chain rule

(a) What happen if the assumption is violated?

If the speed of block propagation is not fast enough, the latest chain information arrives some nodes much earlier than other nodes. They start to assemble next block but others don't. After a period of time, other nodes acquire the out-of-date chain. Then they will create other branches of the chain. If a group of attacker nodes syncs much faster than other nodes, they can assemble illegal blocks leading the chain as long as they are in consensus with each other. It breaks proof-of-work. Therefore, the blockchain never reaches consensus. It never forms the longest chain.

(b) Other assumptions

First is the hash rate of honest nodes is larger than attacker nodes. Attacker controlling majority of hash rate will take up the longest chain, It can assemble any illegal blocks without checking double spending and other necessary verifications.

Second is that none of a single node or organization owns over 51% of the hash rate. Similar with the first assumption, with over 51% of hash rate, it can control the longest chain and control the whole trading market in order to overwrite transactions.

Third is that everyone agrees with POW consensus protocol. Obviously, computation power should be regarded as the only proof of a valid block on the chain. In this way, each honest miner will believe that it is hard for attackers to do 51% attack under POW. Then the legal chain can be continuously extended to the longest one by everyone's mining.

2 What goes wrong without POW

Double spending might occur. Since miners don't need to use computation power to solve hash header puzzle, it is much cheap to commit transactions. The chain is easier to produce a large number of forks. The longest chain is easily replaced by its forks. Then the transactions will be reverted.

Besides, it leads to devaluation of the block rewards because of the low cost of committing transactions. Eventually, no miner wants to commit transactions to the chain if the mining reward cannot cover power cost any more.

3 Transaction & block broadcasting method

Bitcoin is peer-to-peer network. Each node connects maximum 32 peers. The peers are randomly chosen. Let's say account A claims a transaction. A will send it to the surrounding nodes. These

nodes will broadcast the information to their surrounding nodes. Eventually, all of the nodes acquire the transaction information. So is block propagation.

If a new node B wants to join bitcoin network, first B should know one of the nodes address, then the node will send a list of known nodes to B, finally some of them connect B as B's peers.

4 # of blocks to wait

Following bitcoin whitepaper, I implemented a c program as follows:

```
#include <math.h>
#include <stdio.h>
//For (a) question
#define Q 0.2
//For (b) question
//#define Q 0.3
#define Z_MAX 40
double AttackerSuccessProbability(double q, int z){
    double p = 1.0 - q;
    double lambda = z * (q / p);
    double sum = 1.0;
    int i, k;
    for (k = 0; k <= z; k++){
        double poisson = exp(-lambda);
        for (i = 1; i <= k; i++)
            poisson *= lambda / i;
        sum -= poisson * (1 - pow(q / p, z - k));
    }
    return sum;
}
int main(){
    int i = 0;
    for (i = 0; i < Z_MAX+1; i++){
        printf("z = %d, double-spending risk = %lf\n", i, AttackerSuccessProbability(Q,i));
    }
    return 0;
}
```

The c program will print the risk of double spending under different waiting block and attack computation power. `Q` represents attack computation power. `z` is the number of blocks which attackers should catch up.

(a) 0.2 attack computation power, under 0.001 double-spending risk

I set `Q` as 0.2 representing attack computation power. `z` is from 0 to 20. The output from `z = 0` to `20` is as follows:

```
z = 0, double-spending risk = 1.000000
z = 1, double-spending risk = 0.415899
z = 2, double-spending risk = 0.203929
z = 3, double-spending risk = 0.103242
z = 4, double-spending risk = 0.052998
z = 5, double-spending risk = 0.027416
z = 6, double-spending risk = 0.014251
```

```
z = 7, double-spending risk = 0.007432
z = 8, double-spending risk = 0.003885
z = 9, double-spending risk = 0.002035
z = 10, double-spending risk = 0.001067
z = 11, double-spending risk = 0.000560
z = 12, double-spending risk = 0.000294
z = 13, double-spending risk = 0.000155
z = 14, double-spending risk = 0.000081
z = 15, double-spending risk = 0.000043
z = 16, double-spending risk = 0.000023
z = 17, double-spending risk = 0.000012
z = 18, double-spending risk = 0.000006
z = 19, double-spending risk = 0.000003
z = 20, double-spending risk = 0.000002
```

It is found that the risk is lower than 0.001 after 11 blocks. So recipient needs to wait at least 11 blocks.

(b) 0.3 attack computation power, under 0.0001 double-spending risk

I set `Q` as 0.3 representing attack computation power. `z` is from 0 to 40. The output from `z = 20` to `40` is as follows:

```
z = 20, double-spending risk = 0.002480
z = 21, double-spending risk = 0.001875
z = 22, double-spending risk = 0.001417
z = 23, double-spending risk = 0.001072
z = 24, double-spending risk = 0.000811
z = 25, double-spending risk = 0.000613
z = 26, double-spending risk = 0.000464
z = 27, double-spending risk = 0.000351
z = 28, double-spending risk = 0.000266
z = 29, double-spending risk = 0.000201
z = 30, double-spending risk = 0.000152
z = 31, double-spending risk = 0.000115
z = 32, double-spending risk = 0.000087
z = 33, double-spending risk = 0.000066
z = 34, double-spending risk = 0.000050
z = 35, double-spending risk = 0.000038
z = 36, double-spending risk = 0.000029
z = 37, double-spending risk = 0.000022
z = 38, double-spending risk = 0.000016
z = 39, double-spending risk = 0.000012
z = 40, double-spending risk = 0.000009
```

It is found that the risk is lower than 0.0001 after 32 blocks. So recipient needs to wait at least 32 blocks.